AN N LOG N ALGORITHM FOR ISOMORPHISM OF PLANAR TRIPLY CONNECTED GRAPHS

BY

JOHN-. HOPCROFT

STAN-CS-71-192
January, 1971

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

AN N LOG N ALGORITHM FOR ISOMORPHISM OF PLANAR TRIPLY CONNECTED GRAPHS

John Hopcroft

Stanford University

Abstract: It is shown that the isomorphism problem for triply connected planar graphs can be reduced to the problem of minimizing states in a finite automaton. By making use of an n log n algorithm for minimizing the number of states in a finite automaton, an algorithm for determining whether two planar triply connected graphs are isomorphic is developed. The asymptotic growth rate of the algorithm grows as n log n where n is the number of vertices in the graph,

This research was supported by the National Science Foundation under grant number NSF-FJ-96, and the Office of Naval Research under grant number N-00014-67-A-0112-0057 NR 044-402.

AN n log n ALGORITHM FOR ISOMORPHISM OF PLANAR TRIPLY CONNECTED GRAPHS

John Hopcroft

Stanford University

Introduction

The graph isomorphism problem is to determine if there exists a one-to-one mapping of the vertices of a graph onto the vertices of another which preserves adjacency of vertices. At present there is no known algorithm for determining if two arbitrary graphs are isomorphic with a running time which is asymptotically less than exponential. Gotlieb and Corneil [1] have exhibited an efficient algorithm for a large class of graphs, namely those graphs with no k-strongly regular subgraph for large k.

The isomorphism problem for planar graphs is of interest in the study of chemical structures. Weinburg [5] has exhibited an algorithm with asymptotic running time of n^2 for isomorphism of triply connected graphs where n is the number of vertices in the graph. The reason for restricting attention to triply connected graphs is that a triply connected planar graph has a unique representation on a sphere. In this paper we show that isomorphism of triply connected planar graphs can be tested in time proportional to n log n. The algorithm makes use of an n log n algorithm [2] which was developed for minimizing states in a finite automaton. The basic idea is to recognize that minimizing states in finite automaton is really a process of dividing states into equivalence classes. Thus, the algorithm can be applied not only to state minimization but to a wide class of partitioning problems of which the isomorphism of triply connected planar graphs is a member. As a by product of this approach we can associate with each planar triply connected graph a unique reduced graph which in the case of a highly symmetric graph provides a compact encoding of the graph.

Definitions and Notation

A graph G is an ordered pair (V,E) where

(1) V is a finite set of vertices and

(2) E is a finite set of unordered pairs of vertices called edges.

Two vertices u and v are said to be adjacent if the edge (u,v) is in E. Two graphs are said to be isomorphic if there exists a one-to-one mapping of the vertices of one graph onto the vertices of the other which preserves adjacencies. For isomorphism of triply connected planar graphs it suffices to consider only regular degree three graphs with labelled edges. The reason for this is that a vertex of degree $d > 3$ can be expanded into a d-gon and the edges of the d-gon labelled to indicate that they were originally a single vertex. Since the sum of the degrees of the vertices in a planar graph is at most $6n-12$ the number of vertices in the expanded graph is at most $6n-12$.

A finite automaton M is a 5-tuple (S,I,δ,λ,O) where

(1) S is a finite set of states

(2) I is a finite set of input symbols

(3) δ is a mapping of $S \times I$ into S

(4) λ is a mapping of S into O and

(5) O is a finite set of output symbols.

Let I^* be the set of all finite length strings of symbols from I including the empty string ϵ . The mapping δ is extended from $S \times I$ to $S \times I^*$ in the usual manner [4]. Given two finite automata $M_1 = (S_1,I,\delta_1,\lambda_1,O)$ and $M_2 = (S_2,I,\delta_2,\lambda_2,O)$, states q in S_1 and p in S_2 are said to be equivalent if for each x in I^*

$$\lambda_1(\delta_1(q,x)) = \lambda_2(\delta_2(q,x)) .$$

The finite automata M_1 and M_2 are said to be equivalent if for each state q in S_1 there exists at least one equivalent state p in S_2 and vice versa.

Hopcroft [2] has given an algorithm for partitioning the states of a finite automaton into equivalence classes of states. The algorithm can be used to test the equivalence of two finite automata by treating them as a single automaton, partitioning the states, and checking each block in the partition to verify that it contains at least one state from each of the original automata. The asymptotic running time of the algorithm is n log n . Thus we need only show how to associate with each planar triply connected regular degree three graph G , a finite automaton M(G) such that G_1 and G_2 are isomorphic if and only if $M(G_1)$ is equivalent to $M(G_2)$. This will be done in the next section. The conversion time is linear and the number of states in the resulting finite automaton is four times the number of edges in the graph.

Transformation of a Graph to a Finite Automaton

Let G = (V,E) be a regular degree 3, planar, triply connected graph with labelled edges. Assume G is drawn on a sphere. We construct a finite automaton M(G) from G as follows.

$M(G) = (S,\{R,L\},\delta,\lambda,O)$ where

(1) $S = \{[u,v],[v,u] / (u,v) \in E\}$

(2) For each (u, v) in E , $\delta([u,v],R) = [v,w]$ and $\delta([u,v],L) = [v,x]$ where the incident edges at vertex v in clockwise order are (u,v) (v,x) and (v,w) .

(3) $\lambda([u,v]) = [i,j,\ell]$ where i and j are the number of edges in the faces to the right and left of the edge (u,v) when transversed from u to v and where ℓ is the label of the edge [u,v] .

(4) $O = I \times I \times \{$set of labels) .

Intuitively, the states of M(G) correspond to the edges of G along with a direction. If M is in a state corresponding to an edge into vertex v , then on the next input, M will enter the state corresponding to the edge leaving v which is on the right or on the left depending on whether the input is R or L respectively.

Since a planar triply connected graph drawn on a sphere has a parity (that is, left and right depend on whether the graph is viewed from inside or outside the sphere) we define $\hat{M}(G)$ to be M(G) with L and R reversed.

Technical Lemma

This section contains a technical lemma used in the next section. The proof of the lemma is not essential to the understanding of the remainder of the paper.

Lemma 1: Let G be a biconnected planar graph. Let $(v_1,v_2),(v_2,v_3),\ldots,(v_{n-1},v_n)$ be a simple path p in G . Then there exists a face having an edge in common with the path which has the property that the set of all edges common to both the face and the path form a continuous segment of the path. Furthermore, when traversing an edge of the face while going from v_1 to v_n along the path, the face will be on the right.

Proof: If the set of edges common to sane face and to the path consists of at least two discontinuous sets of edges from the path, in both cases the face being on the right of the path, then all faces adjacent to the path from the right between the two sets of edges are adjacent only on the right. Select one such face. Either it satisfies the conditions of the lemma or its edges intersect the path in at least two discontinuous sets of edges. By repeating the process of selecting a face eventually a face satisfying the lemma is selected.

Thus assume that every face which is adjacent to the path on the right is also adjacent to the path on the left. No face can be adjacent to the path on both the right and the left at the same edge since the graph is biconnected. Select a face. Assume that the edge closest to v_n at which the face is adjacent on the right is closer to v_1 than the edge closest to v_n at which the face is adjacent on the left. Then each succeeding face adjacent to the path on the right towards v_n must have the same property. But the face adjacent to (v_{n-1},v_n) on the right cannot have this property. Hence a contradiction. Thus there exists a face satisfying the conditions of the lemma.

Major Result

In this section we show that two planar triply connected graphs G_1 and G_2 are isomorphic if and only if $M(G_1)$ is equivalent to either $M(G_2)$ or $\hat{M}(G_2)$.

Theorem: Let G_1 and G_2 be regular degree three triply connected planar graphs with **labelled** edges. Then G_1 is isomorphic to G_2 iff either

(1) $M(G_1)$ equivalent to $M(G_2)$, or

(2) $M(G_1)$ equivalent to $\hat{M}(G_2)$.

Proof: (only if) Assume G_1 and G_2 are isomorphic. *Clearly* for $M_1 = (S_1, \{R, L\}, \delta_1, \lambda_1, O_1)$ and $M_2 = (S_2, \{R, L\}, \delta_2, \lambda_2, O_2)$, $S_1 = S_2$, $O_1 = O_2$ and $\lambda_1 = \lambda_2$. (We assume that names of corresponding nodes in the two graphs are the same. Thus $S_1 = S_2$.) It remains to be shown that for each q , $\delta_1(q,R) = \delta_2(q,R)$ and $\delta_1(q,L) = \delta_2(q,L)$. Since G_1 and G_2 are triply connected, their representations on a sphere are unique [6]. That is, the order of edges around a vertex is completely specified once a left-right orientation is established. The only if portion follows immediately.

(if) Without loss of generality, assume $M(G_1)$ equivalent to $M(G_2)$. For each state of $M(G_1)$ there exists at least one state *of* $M(G_2)$ equivalent to it. Select a state from S_1 and an equivalent state from S_2 . Since each state corresponds to an edge and a direction, we can identify an edge and a direction in G_1 with an edge and a direction in G_2 . Furthermore, the vertices at the endpoints can be identified. Assume state q has been identified with state p and that the corresponding edges with their respective directions have been identified. Then consider states $\delta_1(q,L)$ and $\delta_2(p,L)$. These two states must be equivalent. We identify the corresponding edges and endpoints. We continue *on* in this fashion always using input L , if possible, to obtain new states to identify. Otherwise we use input R .

The above procedure will eventually map each edge and corresponding endpoints in G_1 to an edge and its corresponding endpoints in G_2 unless a conflict arises. A conflict arises when we try to identify a vertex v_1 in one graph with a vertex v_2 in the other which has already been identified with some $v_3 \neq v_1$. We now prove that if $M(G_1)$ is equivalent to $M(G_2)$, such a situation is impossible.

Assume a conflict arises. Consider the first such instance. One of the edges in the Last pair identified must have completed a cycle. Without loss of generality, assume a cycle was completed in G_1 , Then the corresponding edge in G_2 either did not complete a cycle (the end *vertex* of the edge in G_2 was not previously identified with a vertex of G_1) or it completed *a* different cycle (the end *vertex* of the edge in G_2 *was* previously identified with a *vertex* in G_1 other than the end *vertex* of the edge in G_1) . In the latter *case,* the cycle in G_2 is of different length than the cycle in G_1 . If there are cycles in both graphs, let c be the shorter of the two cycles. If there is a cycle in only one graph let c be that cycle. Let p be the path in the other graph corresponding to the vertices on the cycle c . Note that the first and last vertex-of p correspond to the same vertex in c . Without loss of generality, assume c is in G_1 .

Since there is a cycle in Gl which is mapped to a simple path in G_2 , select that cycle c in G_1 which would map to a simple path p in G_2 but for which no cycle in G_1 other than c containing only vertices from c and its interior would map to a simple path in G_2 . By Lemma 1 some face in G_2 is adjacent to p on the right and all edges of the face which are common to p form a continuous segment of p . Star: identifying the edges around this face in G_2 with edges in G_1 . If a closed cycle *is* completed in

G_1 prior to the canpletion of a closed cycle in G_2 , then there would be a cycle in G_1 containing only vertices **from** c and its interior which would map to a simple path in G_2 a contradiction. If a closed cycle is completed, in G_2 prior to the completion of a closed cycle in G_1 , then a face with, say i edges in G_2 , would map to a path with i edges in G_1 . This is a contradiction since each state has encoded in its output the number of edges in the face, namely i . But in G_1 the face has at least i+l edges. Thus we can assume that both paths are completed simultaneously **and** that two identical faces have been identified. This implies that the vertex at which the path in G_2 terminates was previously identified with the vertex at which the path in G_1 terminates. Now c has been divided into two cycles c_1 and c_2 , Assume cycle c_1 is the face mapped to the face in G_2 . Cycle c_2 is then mapped to a path in G_2 , a contradiction. Since **all** possibilities lead to a contradiction, we are forced to conclude 'that no **conflect** can arise and **that** G_1 and G_2 are indeed isanorphic.

Conclusions

Since the transformation from a **graph** to a finite automaton is such that graphs G_1 and G_2 are **isomorphic** if and only if $M(G_1)$ and $M(G_2)$ are equivalent we can use the state reduction algorithm to test for **isomorphism** of planar triply connected graphs in n log n steps. Note that one need not actually transform the graphs. The state reduction algorithm could be modified to handle graphs directly. It is anticipated that the algorithm will be programmed and this latter approach will be used. Also, it should be noted that there exist algorithms [3] to determine if a graph is three connected in linear time and to determine if a graph is planar in n log n time. The planarity algorithm determines the ordering of the edges about each vertex. Thus, we can start with a list of edges for each graph, rather than the representation on a sphere, and still determine **isomorphism** in n log n steps.

References

[1] Corneil, D. G. and C. C. Gotlieb, "An efficient algorithm for graph isomorphism," JACM 17, (1970), 51-64.

[2] Hopcroft, J. E., "An n log n algorithm for minimizing states in a finite automaton," Technical Report CS-190, Stanford University, Stanford, California. 1970.

[3] Hopcroft, J. E. and R. Tarjan, "Planarity testing in v log v steps: Extended abstract," submitted for publication, Nov. 1970.

[4] Hopcroft, J. E. and J. D. Ullman, Formal languages and their relation to automata, Addison-Wesley, Reading, Mass. 1969.

[5] Weinberg, L., "A simple and efficient algoritbm for determining isomorphism of planar triply connected graphs," IEEE PGEC 13 (1966), 142-148.

[6] Whitney, H., "A set of topological invariants for graphs," Amer. J. Math. 55 (1933), 231-235. .